10-3-2006

An auditory classifier employing a wavelet neural network implemented in a digital design

Jonathan Hughes

AN AUDITORY CLASSIFIER EMPLOYING A WAVELET NEURAL NETWORK IMPLEMENTED IN A DIGITAL DESIGN

by

Jonathan Hughes

A thesis submitted
in
Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Computer Engineering

Approved by

Principal Advisor: _____ Date: _____
Dr. Kenneth W. Hsu

Committee Member: _____ Date: _____
Dr. Pratapa V. Reddy

Committee Member: _____ Date: _____
Dr. Marcin Lukowiak

Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
August 2006

RELEASE PERMISSION FORM

Rochester Institute of Technology

AN AUDITORY CLASSIFIER
EMPLOYING A WAVELET NEURAL NETWORK
IMPLEMENTED IN A DIGITAL DESIGN

I, Jonathan Hughes, hereby grant permission to any individual or organization to reproduce this thesis in whole or in part for non-commercial and non-profit purposes only.

_____
Jonathan M. Hughes

_____
Date

Abstract

This thesis addresses the problem of classifying audio as either voice or music. The goal was to solve this problem by means of digital logic circuit, capable of performing the classification in real time. Since digital audio is essentially a discrete non-periodic time-series, it was necessary to extract features from the audio which are suitable for classification. The discrete wavelet transform combined with a feature extraction method was found to produce such features. The task of classifying these features was found to be best performed by an artificial neural network. Collectively known as a wavelet neural network, the digital logic design implementation of this architecture was effective in correctly identifying the test data sets.

The wavelet neural network was first implemented as a software model, to develop the network architecture and parameters, and to determine ideal results. The unconstrained software simulation was capable of correctly classifying test data sets with greater than 90% accuracy. This model was not feasible as a digital logic design however, as the size of the implementation would have been prohibitive. The size of the resulting hardware model was constrained by reducing the widths of the data paths and storage registers. The hardware implementation of the wavelet processor consisted of a novel pipelined design with a novel data-flow control structure. The neural network training was performed entirely in software by way of a novel training algorithm, and the resulting weights were made to be available to be uploaded to the hardware model.

The digital design of the wavelet neural network was modeled in VHDL and was synthesized with Synplicity Synplify, using Actel ProASICPlus APA600 synthesized library cells with a target clock frequency of 11.025 KHz, to match the sampling rate of the digital audio. The results of the synthesis indicated that the design could operate at 15.6 MHz, and required 96,265 logic cells. The resulting constrained wavelet neural network processor was capable of correctly classifying test data sets with greater than 70% accuracy. Additional modeling showed that with a reasonable increase in hardware size, greater than 86% accuracy is attainable. This thesis focused on classifying audio as either voice or music, and future research could readily extend this work to the problem of speaker recognition and multimedia indexing.

Acknowledgements

There are many people I would like to thank for their contribution to this work, particularly my thesis committee Dr. Kenneth W. Hsu of Computer Engineering, Dr. Pratapa V. Reddy of Computer Engineering, and Dr. Marcin Lukowiak of Computer Engineering.

I would also like to thank Dr. Roger Gaborski of Computer Science Department, Dr. Albert Titus of University at Buffalo, André Botha of Microsoft Corporation, Paul Brown of Intel Corporation, Anne DiFelice of Computer Engineering, and Pam Steinkirchner of Computer Engineering for technical and personal guidance, Stefan Pittner and Sagar V. Kamarthi of Department of Mechanical, Industrial, and Manufacturing Engineering, Northeastern University, for allowing the use of their research in feature extraction.

Finally, I would like to thank my family for their continued support and encouragement: my wife Heather Hughes, father Mr. John Hughes, my mother Mrs. Suellen Hughes, my brother Zachary Hughes, and my sister-in-law Wendy Hughes.

Contents

Abstract ..... **iii**

Acknowledgements ..... **v**

Contents ..... **vi**

List of Figures ..... **viii**

List of Tables ..... **ix**

Glossary ..... **x**

Abbreviations x
Terms xi

Conventions ..... **xiii**

General and Typographical xiii
Diagrams xiii

Chapter 1 Introduction **1**

Part I Review of Literature **2**

Chapter 2 Wavelet Transform Theory **2**

2.1 History of Wavelets 3
2.2 Fourier Analysis 6
2.3 Comparison of Wavelet Analysis to Fourier Analysis 8
2.4 Wavelets in Audio Processing 10

Chapter 3 Feature Extraction Theory **12**

3.1 Time Series Similarity 12
3.2 The Struzik and Siebes Approach 12
3.3 The Pittner and Kamarthi approach 14

Chapter 4 Neural Network Theory **16**

4.1 Perceptron Theory 16
4.2 Multi-Layer Perceptron Neural Network Theory 17

Chapter 5 Wavelet Neural Network Theory **20**

5.1 Overview of Wavelet Neural Networks 20
5.2 Wavelet Neural Networks as Function Approximators 20
5.3 Wavelet Neural Networks as Classifiers 22

Part II Design Implementation of a Wavelet Neural Network **23**

Chapter 6 Design Implementation of a Wavelet Transform **23**

6.1 Software Simulation 25
6.2 Digital Design Implementation 26

Chapter 7 Design Implementation of a Feature Extraction 29
7.1 Software Simulation 29
7.2 Digital Design Implementation 32
Chapter 8 Design Implementation of a Neural Network 34
8.1 Software Simulation 34
8.2 Digital Design Implementation 35
Chapter 9 Design Implementation of a Wavelet Neural Network 40
9.1 Software Simulation 41
9.2 Digital Design Implementation 41
Chapter 10 Simulation Results Comparison between Ideal Software and Digital Design Implementation 46
Part III Conclusion 48
Chapter 11 Conclusion 48
Chapter 12 Future Work 49
Bibliography 50

List of Figures

Figure 1 - Signal and Bus Conventions ..... *xiii*

Figure 2 - Haar Wavelet Function ψ_{00} *4*

Figure 3 - Haar Wavelet dilation and shift *5*

Figure 4 - Comparison of Time and Frequency Resolution *9*

Figure 5 - Haar DWT Processing [9] *24*

Figure 6 - Wavelet Processor *27*

Figure 7 - Low and High Pass Filters *28*

Figure 8 - G_b *Matrix for Feature Cluster Detection* *30*

Figure 9 - Feature Extractor Processor *32*

Figure 10 - Feature Cluster Unit *33*

Figure 11 - Neural Network Processor *37*

Figure 12 - Neuron *38*

Figure 13 - 8x4-bit Multiplier *39*

Figure 14 - Wavelet Neural Network Top Level *43*

Figure 15 - Wavelet Neural Network Components *45*

List of Tables

Table 1 - Feature Cluster Boundaries *31*

Table 2 - Classification Results of WNN Configurations *47*

Glossary

Abbreviations

ω
Angular frequency. Units: radians per second (rad/s).

σ
Population standard deviation.

μ
Population mean.

ASIC
Application Specific Integrated Circuit. A chip designed for a particular application.

DFT
Discreet Fourier Transform. A Fourier transform operating in the discreet domain.

DWT
Discreet Wavelet Transform. A wavelet transform operating in the discrete domain.

FPGA
Field-Programmable Gate Array. A type of logic chip that can be programmed.

g[n]
High-pass filter.

h[n]
Low-pass filter.

Mux
Multiplexor. An analog or digital device that can selectively connect one of a number of inputs channels to an output channel.

O(n)
The Big O notation is a mathematical notation used to describe the asymptotic behavior of functions. More precisely, it is used to describe an asymptotic upper

bound for the magnitude of a function in terms of another, usually simpler, function.

rad

radians (2π rad = 360^o).

SONAR

SOund NAvigation Ranging. An instrument that sends out an acoustic pulse in water and measures distances in terms of the time for the echo of the pulse to return.

VHDL

Very High Speed Integrated Circuit Hardware Description Language. A digital hardware description language used for the modeling and synthesis of digital hardware.

WNN

Wavelet Neural Network. A neural network with a wavelet filter applied to the inputs.

Terms

Actel ProASICPlus

Actel's second generation Flash FPGA family.

Artificial Nueron

The basic unit of an artificial neural network, simulating a biological neuron. It receives one or more inputs, sums these, and produces an output after passing the sum through a (usually) non-linear function known as an activation or transfer function.

APA600

Actel's 600,000 system logic gate FPGA in the ProASIC Plus family.

Bit

Binary digit, the smallest unit of information in a binary notation system and is equal to either a zero (0) or a one (1).

Digital Signal

A signal which is discrete in time and amplitude.

Fourier Transform

A mathematical operation that decomposes a time-varying signal into its complex frequency components (amplitude and phase or real and imaginary components).

Multiplexor

An analog or digital device that can selectively connect one of a number of inputs to an output.

Neural Network

An interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation.

Nyquist Criterion

The minimum frequency at which a continuous bandwidth limited signal can be sampled and the original signal recovered without distortion.

Sampling

the reduction of a signal from continuous time to discrete time.

Synplicity Synplify

Synplicity Corporation's HDL compiler and synthesizer software.

Wavelet Transform

A mathematical operation that decomposes a time-varying signal into its multi-resolution representation that includes both time and frequency localization.

Conventions

Several conventions were adopted in this work and these are presented here.

General and Typographical

Class Naming

C++ Class names are mixed case and italicized e.g. *ClassName.*

Signal Naming

Signals are denominated using uppercase letters and appear in the text in typewriter font e.g. `SIGNAL`.

Diagrams

Block Diagrams

Signals are displayed in block diagrams as arrows, and signal busses are displayed in block diagrams as arrows with a number superimposed on the arrow, which represents the bus width, as shown in Figure 1.

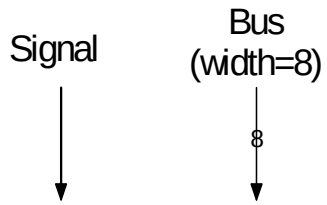

Figure 1 - Signal and Bus Conventions

Chapter 1 Introduction

With the advent of neural network processing, computer systems have been able to replace, and in some cases improve upon human operators. One such system would be auditory processing. Auditory processing is being developed to solve problems such as voice recognition, speaker recognition, multimedia indexing, SONAR analysis, and many others [1]. In order to present audio samples to a neural network, the important features of the sample must first be extracted. Discreet wavelet transform processing combined with feature extraction techniques have been shown to be effective in this task [2]. The proposed system incorporates a discreet wavelet transform processor, which employs a novel hierarchical filtering architecture, as well as a novel data control system for loading and propagating audio samples and wavelet coefficients, a feature extraction processor, and a neural network, which was trained with a novel algorithm, to classify audio samples as belonging to either a voice class or music class. The system was modeled in a digital design using VHDL as the hardware description language.

Part I Review of Literature

Chapter 2 Wavelet Transform Theory

Signal analysis has benefited from mathematical tools, such as the Fourier transform and more recently, wavelet transforms, which manipulates the signal into a more usable form for analysis. The principal of superposition has been used to approximate functions since the early 1800's, when Joseph Fourier discovered that any periodic function could be represented by superimposing sine and cosine functions. Since Fourier's transform relies on the use of non-local periodic functions to approximate target functions, attempting to represent local non-periodic functions with discontinuities and sharp spikes results in a less accurate approximation. The wavelet transform is effective in approximating such functions, as it uses local non-periodic basis functions in its analysis [3].

Another key component of the wavelet transform is its multi-resolution analysis. If examining a function using a small "window", small details of that function within that window would be readily apparent. If examining the same function with a large window, the gross features of the function would be revealed. The multi-resolution analysis of the wavelet transform approximates the function in such a way that both the small details are represented, as well as the gross features, and all scales in between [3].

Wavelet analysis is accomplished by first choosing a representative prototype function, called the mother wavelet, or analyzing wavelet. The target function is then approximated by using contracted and dilated versions of the prototype function. The contracted, higher frequency versions result in an increased capacity for temporal

analysis, and decreased frequency analysis. The dilated, lower frequency versions of the prototype result in an increased capacity for frequency analysis, and decreased temporal analysis. Together, these contracted and dilated prototypes perform an analysis that gives a complete representation of the target function, both in the time and frequency domains [3].

2.1 History of Wavelets

Joseph Fourier, the father of Harmonic Analysis, began a field of mathematics, which eventually lead future mathematicians to Wavelet Analysis. Fourier, in 1807, postulated that any periodic function *f(x)* can be expressed as the sum

$$a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx) \tag{1}$$

of its Fourier series. The coefficients, a_0, a_k, and b_k, are calculated by

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx, \quad a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)dx, \quad b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)dx$$

For the remainder of the 19th century, researchers investigated the meanings and applications of Fourier's frequency analysis, which led those of the early 20th century to explore the concept of scale analysis. Scale analysis involves constructing a function, and then scaling and shifting it, and then applying it to a target function to obtain an approximation. This process is then repeated by shifting and scaling until a complete approximation of the target function is obtained [3,4].

Wavelets were first mentioned in the appendix of Alfred Haar's thesis in 1909. Haar showed that any continuous function f(x) on the unit interval [0,1], could be approximated by a series of step functions. The Haar wavelet is the simplest possible wavelet which is a step function with the definition

$$f(x) = \begin{cases} 1 & 0 \le x < \frac{1}{2}, \\ -1 & \frac{1}{2} \le x < 1, \\ 0 & otherwise \end{cases} \tag{2}$$

and shown in Figure 2. Figure 3 shows how the Haar mother wavelet ψ_{00} is dialated by narrowing its support, and shifted along the unit interval. The individual elements are linearly independent and can represent step functions of increased thinness.

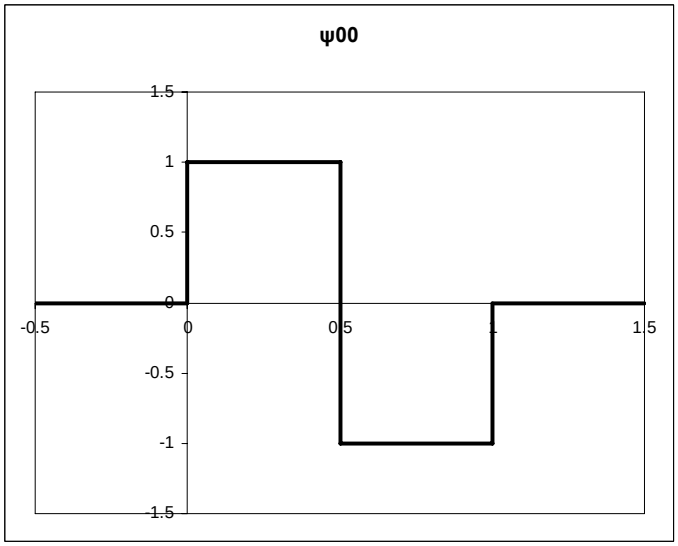

Figure 2 - Haar Wavelet Function ψ_{00}

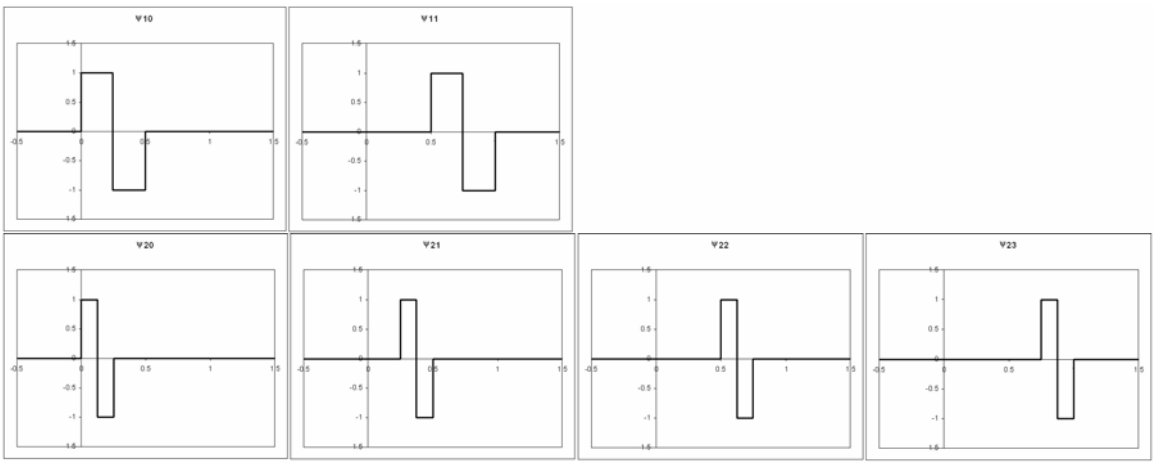

Figure 3 - Haar Wavelet dilation and shift

The Haar wavelet has compact support, because it disappears beyond a finite interval, but because it is not continuous, it is also not differentiable, which limits some of this wavelet's application [3].

Wavelet research continued since the 1930's to include an application of Haar's wavelet in the study of Brownian motion by Paul Levy, who found it superior to Fourier functions in studying the small details of the motion. In the latter half of the twentieth century, Guido Weiss and Ronald R. Coifman analyzed a function space by reducing it to its "atoms", and defined assembly rules for these most primitive elements. Jean Morlet and Alex Grossman [3], whom originally worked windowed Fourier analysis, further developed wavelet theory by discovering that a wavelet transform could be reversed losslessly to restore the original function, and that any changes to the wavelet coefficients resulted in very small changes to the reverse transformed signal. Stephane Mallat [3] brought wavelets in to mainstream acceptance with his work on quadrature mirror filters

and orthonormal wavelet bases. He further developed an iterative approach to the wavelet transform with a filter which employs a pyramid algorithm, laying the foundation for the Fast Wavelet Transform (FWT), the wavelet correspondent to the Fast Fourier Transform (FFT). Yves Meyer [3] created the first non-trivial wavelet, which was continuously differentiable, but lacked compact support. Ingrid Daubechies has the honor of having created the most elegant wavelet function that has become the cornerstone of wavelet applications today. The Daubechies families of wavelets are both continuously differentiable, and have compact support [3,5].

2.2 Fourier Analysis

Fourier's concept of the superposition of sine and cosine to approximate functions has had relevant application ranging from the pure mathematical of solving differential equations to the applied engineering of analyzing communication signals [3].

The Fourier transform's value is in its ability to allow a periodic function in the time domain to be analyzed for its frequency content. This is accomplished by first transforming the time domain function into a function in the frequency domain. The Fourier coefficients generated in this process directly correspond to the contribution of the associated sines and cosines at each frequency. The inverse Fourier transform converts a function from the frequency domain, back into the time domain [3].

The Discrete Fourier Transform (DFT) approximates the Fourier transform of a function from a finite number of discrete samples, which are representative of the remainder of the function. The discrete Fourier transform shares many of the symmetry properties of the continuous transform, and also has the added benefit of being able to use the same

calculation to compute the discrete Fourier transform as the Inverse Discrete Fourier Transform (IDFT).

$$\text{DFT:} \quad X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0,\ldots,N-1 \tag{3}$$

$$\text{IDFT:} \quad x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \qquad n = 0,\ldots,N-1 \tag{4}$$

The Fourier transform does an excellent job of representing periodic continuous functions with sines and consines. It struggles, however, to represent non periodic or discontinuous functions. One approach would be to extend the function to attempt to make it periodic, but one would also need to make the endpoints continuous. For this reason, the Windowed Fourier Transform (WFT) was created to address this problem. The WFT chops up a signal into time slices, and then analyze the frequency content of the function bounded by its window. The endpoints of the function at the window boundaries could still present a problem to the Fourier transform however, so a weighting system is used to bias the endpoints toward a value of zero. The WFT also has the added benefit of localizing the frequency analysis in time [3].

In the case of using the discrete Fourier transform to approximate the Fourier integral of the sampled function, the DFT equation (3) requires a calculation of the order O(n^2) for a function of n samples. The problem quickly grows in complexity as the sample size increases. For this reason, it is important to discuss the valuable contributions of J. W. Cooley and J. W. Tukey in 1965, who proposed a solution to the DFT which only required calculations of the order O(n log n) [6]. Cooley and Tukey's solution involves

recursively dividing the calculation from a DFT of size $N = N_1N_2$ into separate DFT calculations each of size N_1 and N_2 respectively. Their work resulted in the so called fast Fourier transform (FFT) [3].

2.3 Comparison of Wavelet Analysis to Fourier Analysis

The wavelet transform shares many similarities to the Fourier transform. For example both the Discreet Wavelet Transform (DWT) and the Fast Fourier Transform (FFT) are linear operations which generate $\log_2$ n vectors of varying lengths, which when combined form a vector of 2^n total length. The similarities also extend to the matrices involved in these calculations. The inverse transforms for both the DWT and FFT use the transpose of the matrix from the forward transform. Therefore, each transform can be considered a rotation of function space to a different domain. Both transforms are also dependent on the use of basis functions; the FFT uses sine and cosine basis functions, and the DWT uses a more complicated basis function called a wavelet, a mother wavelet, or an analyzing wavelet. These basis functions are localized in frequency, both for the FFT and the DWT, which makes them useful for such calculations as power spectra [3].

The dissimilarities between the wavelet transform and the Fourier transform give the wavelet transform advantages in many types of time series analysis. The most significant dissimilarity is that wavelet functions are localized in space, whereas the sines and cosines of the Fourier transform are not. Wavelet function's time localization combined with its frequency localization give functions transformed into the wavelet domain a sparse nature, which makes them particularly useful for feature detection, removing noise from a time series, data compression, and other applications [3].

One way to visualize the difference between the time-frequency resolution of the Fourier transform and the wavelet transform is to examine the basis function coverage of the time-frequency plane. In Figure 4, the time-frequency coverage is compared between the Fourier transform, the windowed Fourier transform, and the wavelet transform.

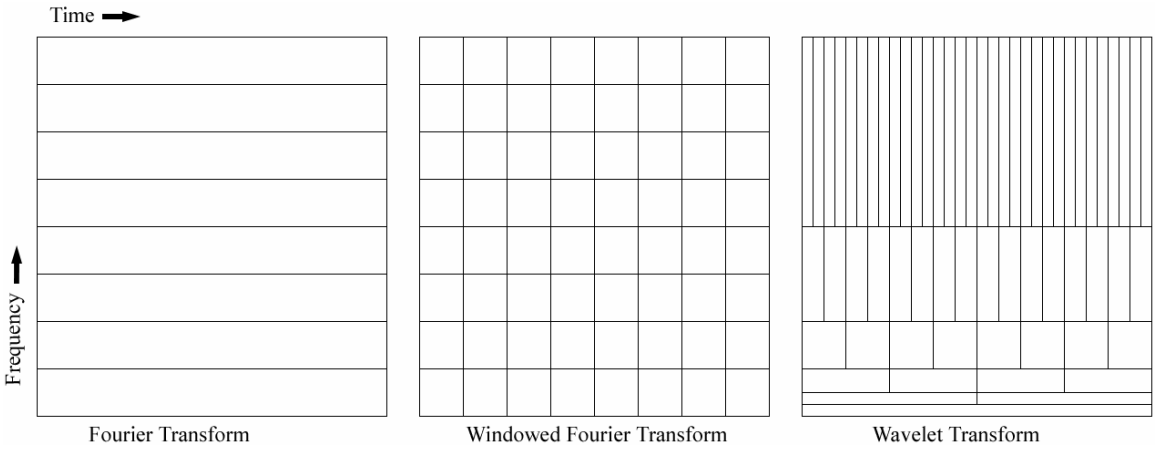

Figure 4 - Comparison of Time and Frequency Resolution

It can easily be seen from Figure 4, that the Fourier transform has no time resolution, and only frequency resolution. This is because the sine and cosine basis functions are infinite in length, and only change in frequency. The windowed Fourier transform shows the same time resolution at all frequencies, because the size of the window in the WFT is the same at all frequencies, and truncates the sines and cosines [3].

With the wavelet transform, the time resolution changes at each frequency analyzed. For analysis of small discontinuities, it would be beneficial to have some very short basis function, but for detailed frequency analysis, very long basis functions would be required. Wavelets capture both of these qualities by having very short high frequency basis

functions as well as very long low frequency basis functions, as shown in Figure 4. An important quality of the wavelet transform is that there is an infinite set of different basis functions to choose from, where as Fourier transforms utilize only the sine and cosine basis functions. Wavelet analysis provides direct access to information that might otherwise be obscured by other time-series analysis methods such as the Fourier transform [3].

2.4 Wavelets in Audio Processing

In the case of digital audio processing, a discreet transform can be used in the analysis of a digital signal. Discreet Wavelet Transform (DWT) analysis is of greater benefit to audio processing than Discreet Fourier Transform (DFT) analysis, because DWT analysis is a multi-resolution representation that includes both time and frequency localization, whereas DFT analysis only provides frequency localization [7]. Features generated, from time and frequency information, are of greater value to audio analysis than are features generated from frequency information alone, because this closely approximates human auditory processing [8].

A wavelet is a mathematical function, which, when applied to a signal, generates a multi-resolution time-frequency representation of that signal. This is similar to a sub-band encoding, in which each successive sub-band has incrementally more time representation, and incrementally less frequency representation. The time-frequency components are generated by sets of scaling functions and wavelet functions, which are associated with low-pass and high-pass filters, respectively. The time domain signal is successively filtered by both the low-pass and the high-pass filters, and the outputs of which, are then

down-sampled. After each stage of low- and high-pass filtering, the resulting half bands contain the same number of samples as the input, but half the frequency. According to the Nyquist Criterion, half of the resulting samples can be discarded without loss of information. The down-sampled outputs of the high-pass filter of each stage become wavelet detail coefficients, and the down-sampled outputs of the low-pass filter are again processed by the low- and high-pass filters of the next stage, with the final stage outputs of the low-pass filter being the wavelet approximation coefficients. The combination of the approximation coefficients and the detail coefficients from each stage becomes the DWT coefficients, or wavelet coefficients [9, 10].

Since the low-pass filter effectively removes half of the frequencies, the resulting signal has twice the frequency resolution of the input signal. However, since half of the resulting samples are discarded due to down-sampling, the time resolution is halved. Therefore, the successive filtering and down-sampling produces coefficients with a greater frequency resolution, and a lesser time resolution after each stage of the process. Therefore, the generated detail coefficients of each stage have increased frequency resolution, but reduced time resolution. Unlike DFT, the time localization of the signal is maintained as well as the frequency localization, however it localized within a resolution. Signal analysis often requires a multi-resolution approach, which makes this scheme an attractive option [9, 11].

Chapter 3 Feature Extraction Theory

3.1 Time Series Similarity

The problem of data comparison is often fairly trivial for most data domains. One must merely determine the similarity or equality of one or more attributes for a datum to compare against another in the same given domain. This procedure is not inherently extensible to comparing one time series to another, because there are no inherent attributes of a time series with which to compare [12]. The problem of generating attributes for a time series with which to compare has been a topic of recent interest, and several approaches have been proposed. This chapter covers two such approaches, the latter of which was implemented in this thesis. The Struzik and Siebes approach [13] to generating attributes for a time series results in features that can be directly compared via a quadratic distance measure. The Pittner and Kamarthi approach [2] to generating attributes for a time series results in features that are designed to be directly processed by a neural network.

3.2 The Struzik and Siebes Approach

When comparing time series, one would want to choose a comparison that ignores bias, both constant and linear, and also ignores scale, both in time and amplitude. Subjective, qualitative judgments of similarity (by humans) are based on non-stationary behavior, rapid transients that indicate beginnings of trends, large fluctuations, and rare events. Whereas a statistical estimation of a time series would be thwarted by such fluctuations, these statistical deviations can be measured and transformed into features for time series comparison [13].

Struzik and Siebes found that by applying the wavelet transform to the time series the local behavior of the function is revealed, as well as the above mentioned biases are suppressed. They also found the wavelet transform to be excellent at characterizing non-stationarities, as well as to reveal the hierarchy of singular features including their scale free behavior. Using Mallat's [14] representation of the wavelet transform, the Wavelet Transform Modulus Maxima (WTMM), they could characterize local singular behavior of time series. The WTMM can also be used for the evaluation of the Hölder exponent of the singularity:

$$W^{(n)} f(s, x_0) \approx |s|^{h(x_0)}, \tag{5}$$

if $h(x_0) < n + 1$, where n indicates the number of vanishing moments of the wavelet. The Hölder exponent can be used as a representation of the roughness or smoothness of the time series, where larger values of the Hölder exponent relate to smoother, more regular time series [13].

Struzik and Siebes' approach to generating attributes for a time series involves including a certain number of predefined features. These features are obtained by taking the predefined number of strongest maxima, and then tracing them below the representation scale from which they appear. This method allows for better localization of singularities in the time domain, as well as to gain a more stable estimation of the Hölder exponent. For a given maxima, the datum recorded has the following attributes: the x-coordinate x_i, the Hölder exponent $h(x_i)$, and the corresponding sign of the wavelet transform $Wf(x_i)$. These features can then be compared by means of a quadratic distance measure, with separate factors for position and h exponent, f_x and f_h respectively:

$$dist_s(x,h) = 1 - (f_x \Delta_x{}^2 + f_h \Delta_h{}^2), \quad (6)$$

where $\Delta_x = x - x_i$ and $\Delta_h = h - h_i$ and x_i,h_i belong to s, the representation of the time series [13].

3.3 The Pittner and Kamarthi approach

The data provided by the wavelet decomposition process, though multi-resolution and time- and frequency-localized, is not of an acceptable form for neural network processing. The wavelet coefficient data set is overly large, and includes data that could be not meaningful or could be misleading to the training or classifying process of neural networks. A feature extractor processor, to generate meaningful features from the wavelet coefficients, is required. The feature extraction process described by Stefan Pittner and Sagar V. Kamarthi [2] was chosen for this thesis.

The feature extraction process requires initial setup steps that involve processing a training set of data. This training set is the same data set that will be used to train the neural network. The use of the training set allows the feature extraction process to be customized for the data space of interest. The feature extraction process depends on the use of clusters to identify groups of wavelet coefficients. The determination of the number of clusters and the size of each cluster is accomplished by the following procedure. First, the coefficients are placed in a matrix, *B*, such that each row vector contains all of the detail coefficients for a given level of decomposition, and the last row vector contains the approximation coefficient. All empty spaces in the matrix are filled with zeros. For each data set in the training set, a matrix, *B*, is formed and is inserted into

an array of matrices, B_K, where K is the length of the training set. Let I be the matrix of the same size as B but containing only 1's as its elements. Let R be the operation that reduces a matrix by its last row. A matrix, G, is calculated by the following equation.

$$G = (g_{ij}) \coloneqq \frac{1}{\sigma\left(R\left(\sum_{k=1}^{K} B_k\right)\right)}\left(\sum_{k=1}^{K} B_k - \mu\left(R\left(\sum_{k=1}^{K} B_k\right)\right) \cdot I\right) \cdot \tag{7}$$

By applying a threshold of the form $T \coloneqq \sqrt{2(\ln L - \ln \gamma)}$ with $\gamma \geq e^2$ to the elements of the matrix G, with L being the number of computed detail coefficients, the binary matrix,

$$G_b \coloneqq \left(\Theta\left(g_{ij} - T\right)\right), \tag{8}$$

is obtained, where the Heavyside function $\Theta(x) = 1$ for $x \geq 0$ and $\Theta(x) = 0$ for $x < 0$ [7]. The binary matrix, G_b, contains 1's at the center of the proposed clusters. If a row vector of G_b contains no 1's, the entire row vector is treated as a cluster. This ensures that clusters do not overlap across different scales.

Extracting the features of the wavelet coefficients involves using the clusters that were generated. For each cluster, U, the feature, u, is calculated by taking the square root of the sum of the squares of each coefficient, v, in the cluster, also known as the Euclidean norm [2].

$$u_i \coloneqq \|r_i\|_2 = \sqrt{\sum_{v \in U_i} v^2} \tag{9}$$

Chapter 4 Neural Network Theory

4.1 Perceptron Theory

Neural networks can be applied to solve classification problems by means of a learning process. Using the learning process, the solution to the classification problem can be found without the need for complex, often slow and inaccurate, algorithms. There are varieties of different types of neural networks that have been proposed by researchers. One of which, is the multi-layer perceptron, which has its basis in neural biology [15]. The multi-layer perceptron has been found to be the classic solution in the task of classifying signal based problems [16, 17].

The basic building block of the multi-layer perceptron is the artificial neuron, also known as the perceptron. The McCulloch-Pitts model of a neuron [15] is represented by synapses, an adder, and an activation function. The synapses are the inputs to the neuron, each with its own weight in order to adjust the strength of the input. The adder component combines the inputs of the neuron, and multiplies each by its respective weight. This weighted sum is called the activation potential. The activation function then applies a "squashing function" to the activation potential. This limits the permissible amplitude range of the output signal [15]. For linearly separable classes, a simple threshold function would suffice for the activation function, such as

$$\phi(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}. \tag{10}$$

The model of a perceptron, considered for this thesis, is based on the McCulloch-Pitts model of a neuron. The goal of the perceptron is to correctly classify a set of inputs. In the simplest form of the perceptron, there are two decision regions separated by a hyper-plane. Therefore, these classes must be linearly separable. The perceptron is trained with a known training set, which must be limited to two linearly separable classes. The perceptron converges using an error back propagation learning algorithm [15].

4.2 Multi-Layer Perceptron Neural Network Theory

The model of the multi-layer perceptron, considered for this thesis, is based on the single perceptron. The goal of the multi-layer perceptron is to correctly classify a set of inputs within an input space that is defined by more than two decision regions, and is therefore non-linear.

The activation function must be a non-linear function, in order that the perceptron can classify patterns that are not linearly separable. For non-linearly separable classes, a sigmoid function is traditionally chosen because of its simple derivative. Examples of sigmoid functions include the special case of the logistic function, where

$$\phi(x) = \frac{1}{1+e^{-x}}, \tag{11}$$

and the hyperbolic tangent, where

$$\phi(x) = \tanh(x) \text{ [18].} \tag{12}$$

This thesis employs the following theoretical model of the neuron. The neuron function first computes the activation potential, v_k, from the following equation.

$$v_k = \sum_{j=0}^{m} w_{kj} \cdot x_j \tag{13}$$

The activation function for each neuron is defined to be

$$\phi_k(v_k(n)) = a \cdot \tanh(b \cdot v_k(n)) \tag{14}$$

Values for *a* are chosen to allow for adequate buffering between the desired response and the actual response. Values for b are chosen to allow for a larger range of *vk* values, in order to avoid railing the actual response.

The number of layers and the number of neurons in each layer determine the number of decision regions that a multi-layer perceptron can define. A typical multi-layer perceptron consists of an output layer, and one or more hidden layers, one of which is also known as the input layer. The multi-layer perceptron is first trained with a known training set. This is accomplished by applying the known input to the input layer, and then forward propagating the results through the other layers. During this phase, the weights remain constant. The results from the output layer are then collected, and compared to the desired response, which is determined from the known input. An error signal is calculated from the difference of the actual response and the desired response. This error signal is then back propagated through the neural network, against the direction of the synaptic connections. The multi-layer perceptron converges using a back-propagation error-correction learning algorithm [15].

After each training iteration, the weights in the neural network are modified according to the following back-propagation error-correction equations,

$$\Delta w_{ji}(n) = \eta \cdot \delta_j \cdot y_i(n), \tag{15}$$

where η is the learning rate, and y_i is the neuron output value.

If j is in the output layer,

$$\delta_j(n) = \left(d_j(n) - y_j(n)\right) \cdot \frac{b}{a} \cdot \left(a - y_j(n)\right)\left(a + y_j(n)\right), \tag{16}$$

or if j is in a hidden layer,

$$\delta_j(n) = \frac{b}{a} \cdot \left(a - y_j(n)\right)\left(a + y_j(n)\right) \cdot \sum_k \delta_k(n) \cdot w_{kj}(n), \tag{17}$$

where d_j is the desired response, and a and b are the scaling values from the neuron activation function. Training continues until the weights of the neural network produce outputs that converge. Convergence is defined by an average error signal, ε_{av} reaching a threshold.

$$e_j(n) = d_j(n) - y_j(n) \tag{18}$$

$$\varepsilon(n) = \frac{1}{2} \cdot \sum_{j \in c} e_j^2(n), \tag{19}$$

where c is the set of all neurons in the output layer.

$$\varepsilon_{av} = \frac{1}{N} \cdot \sum_{n=1}^{N} \varepsilon(n) \tag{20}$$

Chapter 5 Wavelet Neural Network Theory

5.1 Overview of Wavelet Neural Networks

Applications of Wavelet Neural Networks (WNN) are nearly as varied as their possible configurations. Two popular applications of WNNs are their use as function approximators, and as signal classifiers, the latter of which is the focus of this thesis. In the domain of time series analysis, Wavelet Neural Networks improve upon the classic Artificial Neural Network (ANN). ANNs have limited ability to characterize local features of a time series, which are generally critical to accurately classifying or modeling the series. Since these features are often localized in time and/or frequency, employing wavelets enables the Neural Network to take advantage of the multi-resolution analysis offered by wavelets to focus the network on these local features [19]. Generally speaking, a wavelet function is used to condition the inputs to the neural network, such that only vital information about the signal is processed by the network. When designing a Wavelet Neural Network, the possible configurations of wavelet functions, neural network architectures, and their integrations, are nearly limitless.

5.2 Wavelet Neural Networks as Function Approximators

The use of Wavelet Neural Networks as function approximators lends to applications involving process modeling, non-linear function approximation, non-parametric estimation, control tasks, time series forcasting, and many others [19]. As opposed to classical ANNs which use sigmoidal-based activation functions, WNNs of this type typically employ the Inverse Discrete Wavelet Transform (IDWT) as the activation

function for the hidden layer neurons, and a linear output layer which represents the weighted sum of the hidden layer.

Each neuron in the hidden layer represents a wavelet coefficient, such that its corresponding IDWT is localized in time and frequency with a given dilation and translation [20]. Since the wavelet transform results in a sparse representation, not all of the wavelet coefficients are necessary for an accurate reconstruction of the original signal [21]. In fact, the inclusion of all of the coefficients would likely cause over training of the neural network, and result in poor convergence. For this reason, wavelet coefficients that do not contribute to the local features of the signal are identified during the iterative training of the WNN, and their corresponding neurons are pruned from the network [22].

As an application of this technology, Geva discusses a modification to this architecture called ScaleNet [23]. ScaleNet is a multiscale Neural Network architecture for time series prediction. For history based prediction to succeed, information must be present in the history of the time series that indicates the future of the time series. Geva's approach to this problem results in a three stage architecture consisting of a wavelet analysis of the history of the time series, which feeds into several independent feed-forward neural networks, each responsible for one scale of the wavelet decomposition. The results of each neural network, wavelet coefficients of the predicted time series increment, are then propagated to a final stage perceptron which combines each weighted contribution to the prediction.

5.3 Wavelet Neural Networks as Classifiers

The use of Wavelet Neural Networks as classifiers lends to applications involving machine failure mode analysis, and auditory processing tasks, including voice recognition, speaker recognition, multimedia indexing, SONAR analysis, and many others. WNNs of this type are typically found in one of two main configurations. One of these configurations involves a neural network which uses wavelets as activation functions, such that each neuron implements the DWT at a different dilation and translation.

Wang, Yu, and Lee [24] discuss the application and implementation of a WNN of this configuration. The application of their WNN involves machine failure mode analysis, such that sensors on the machine provide the WNN a steady time series of information. An analysis of the time series can reveal the health of the machine, from normal, to degraded, to failure. The neural network was trained with widowed samples of each of these states, resulting in a WNN capable of indicating to an operator when the machine is in the degraded state, before it fails [24].

The other configuration of WNNs used for classification, such as the one modeled in this thesis, involves a wavelet preprocessor, feature extractor, and a classic sigmoidal-based ANN to classify the time series. A window of the time series is converted to wavelet coefficients by the wavelet preprocessor, and then features are generated from the wavelet coefficients, which are then fed to the input (hidden) layer of the ANN. This configuration of WNN can be trained to classify the windowed time series through standard ANN back-propagation error-correction techniques.

Part II Design Implementation of a Wavelet Neural Network

Chapter 6 Design Implementation of a Wavelet Transform

The Haar wavelet transform was chosen as the DWT for this thesis because it lends well to a digital design, and it is the simplest of the wavelet transforms, yet has been proven effective in signal analysis. The Haar transform can be implemented rather easily, it requires minimal intermediate storage, and it's computation is bounded by order O(n), which allows for a pipelined digital design [7]. Conceptually, the Haar wavelet transform is a recursive filtering, beginning first with the original sample vector, which is processed by both a low-pass and a high-pass filter. The output of the high-pass filter is collected to the result vector as detail coefficients, and the output of the low-pass filter is successively filtered as previously described. This repeats until each filter generates only one coefficient. The output of the final low-pass filter is the approximation coefficient. This process is depicted graphically in Figure 5 [9].

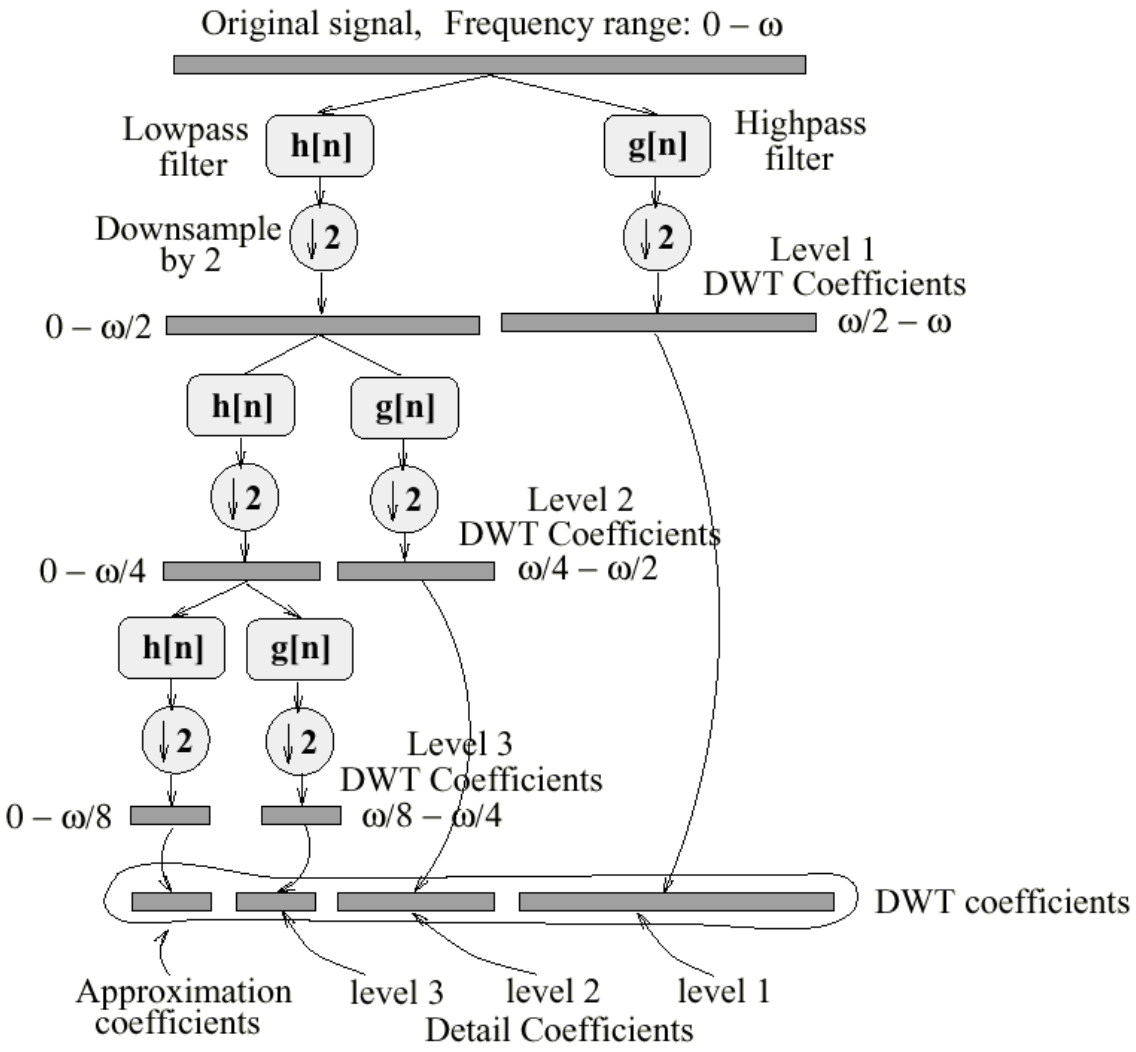

Figure 5 - Haar DWT Processing [9]

The low-pass filter produces outputs that are the sum of adjacent inputs divided by two, and the high-pass filter produces outputs that are the difference of adjacent inputs divided by two. The outputs of each filter are of the same number as the inputs, but have half the frequency band. Therefore, half of the outputs can be discarded, according to the Nyquist Criterion. Thus, the transform process can be performed in $\log_2(n)$. stages, where n is the number of input samples. This transform produces (n-1) detail coefficients, and one

approximation coefficient. If the process were performed in reverse, the original signal could be reconstructed; therefore, there is no loss of information in this procedure [9]. A frame size of 256 input samples was chosen for this thesis, which has been determined to be an effective, yet minimal size [16].

6.1 Software Simulation

The design of the wavelet transform processor for the software simulation was fairly straightforward. The *Wavelet* class accepts a 256-entry array of 16-bit audio samples as inputs, and generates a 256-entry array of 4-bit wavelet coefficients. The *Wavelet* class consists of a process function which calls a filter function on the input samples and then successively on the resulting filtered outputs. The filter function in turn calls both a high- and a low-pass function. For each of the adjacent pairs of the input set, the high-pass function computes the difference of the pair, and divides the difference by two. For each of the adjacent pairs of the input set, the low-pass function computes the sum of the pair, and divides the sum by two. The outputs of the high-pass filter, the detail coefficients, are collected in a result array, and the outputs of the low-pass filter are propagated to the filter function of the next processing stage. This procedure continues until the high- and low-pass filters produce a single output. The final output of the high-pass filter is stored in second position of the result array, and the final output of the low-pass filter, the approximation coefficient, is stored in the first position of the result array. Following the filtering operations, the coefficients are converted to 4-bit values by means of shifting and sign-extending.

6.2 Digital Design Implementation

A novel approach was taken in the design of the wavelet transform processor for this thesis. The wavelet transform processor was designed to accept uncompressed digital audio, sampled at 11.025 kHz at 16 bits per sample. The digital audio is processed in blocks of 256 samples or approximately 23 milliseconds. This requires $log_2(256) = 8$ levels of filters to transform the audio data into wavelet coefficients. Instead of designing an expansive tree of filters and intermediate registers to simultaneously perform the transform operation on the entire block of 256 samples, a novel approach was used in the design to pipeline the samples through 8 levels of low- and high-pass filters and intermediate capture registers. An intricate control system was designed to control the loading and propagating of the data at the intermediate capture registers and the result registers. Due to synchronization issues, the process takes more than 256 cycles to complete. Therefore, an additional pipeline stage of temporary save-registers were used to prevent the overwriting of the results by the processing of the next block of samples. The wavelet transform processor's outputs are 256 4-bit wavelet coefficients. A block diagram of this system is shown in Figure 6. A block diagram of the high and low pass filters is shown in Figure 7.

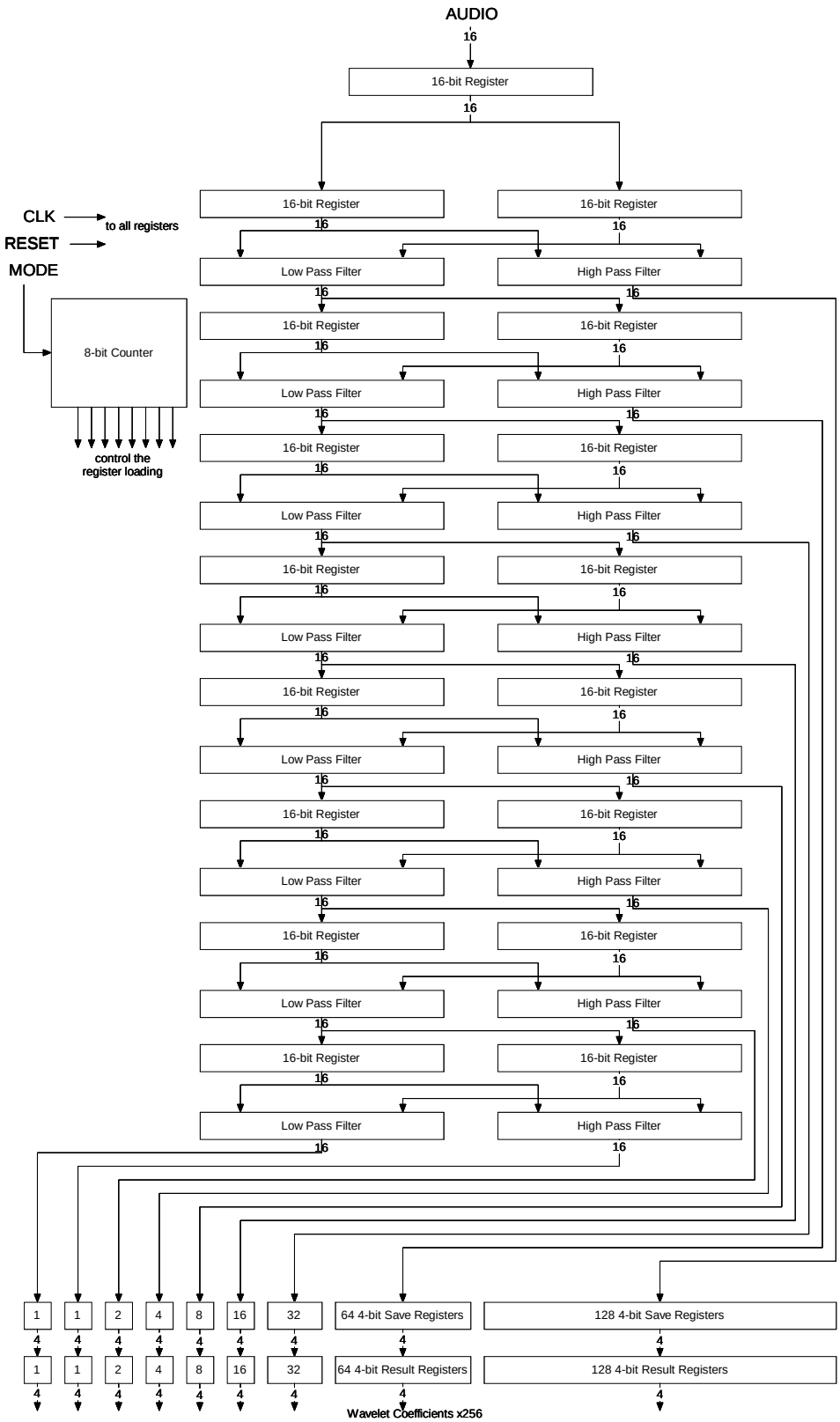

Figure 6 - Wavelet Processor

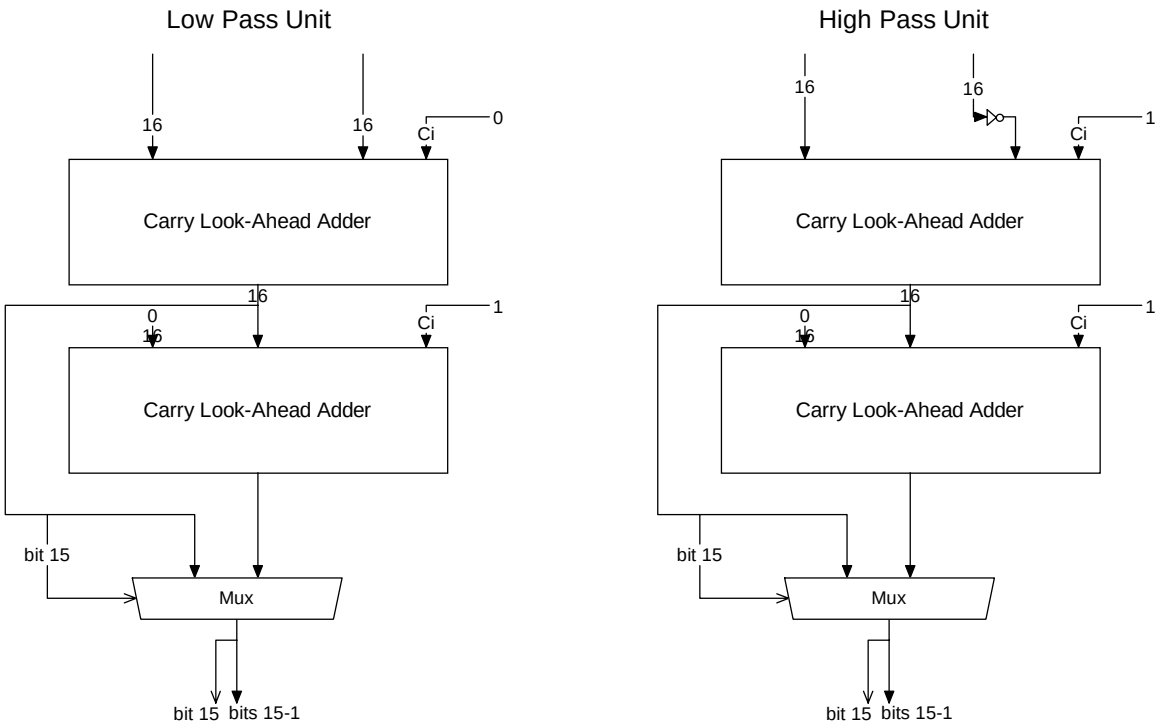

Figure 7 - Low and High Pass Filters

Chapter 7 Design Implementation of a Feature Extraction

The design of the feature extractor processor closely follows the feature extraction process described previously. In a software development environment, the training set of audio samples was transformed by the wavelet process into wavelet coefficients. The feature cluster analyzer software, as described above, then processed these wavelet coefficients. This resulted in discovering 34 clusters for the data space considered in this thesis.

7.1 Software Simulation

The feature cluster analyzer software was designed to take the training set of wavelet coefficients, and calculate the boundaries of the feature clusters. This was accomplished by means of software to calculate the G matrix, and a spreadsheet to calculate the G_b matrix.

The software to calculate the G matrix, read files containing the stored voice and music wavelet coefficients from the training set, and constructed a B matrix for each set of 256 coefficients. These B matrices were placed in an array B_k, which was then used to calculate the G matrix, in a process, described in equation (7). The G matrix was then stored as a file.

The spreadsheet used to calculate the G_b matrix was created from the stored G matrix file. The threshold T was then subtracted from each element of the G matrix. The threshold T was calculated to be 2.6613, by using the equation, $T := \sqrt{2(\ln L - \ln \gamma)}$, where L is the number of detail coefficients, which is 255, and γ is greater than or equal to e^2, which was chosen to be e^2. The Heavyside function was then applied to each element of the intermediary matrix, as described in equation (8), which resulted in the G_b matrix. For illustrative purposes, the resulting G_b matrix is depicted graphically in Figure 8. Black squares represent 1's in the binary matrix, and white squares represent 0's. Gray squares represent areas of the matrix that have no corresponding wavelet coefficient. The bottom left square represents the approximation coefficient, and the remaining non-gray squares represent the eight levels of detail coefficients.

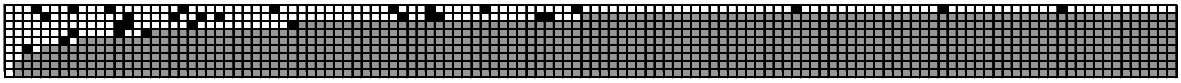

Figure 8 - G_b Matrix for Feature Cluster Detection

The feature clusters were determined by using each "1" in the matrix as the approximate center of the cluster. If a matrix row had no 1's, the entire row was designated as a cluster. This ensured that clusters did not overlap across different scales. The resulting cluster boundaries are shown in Table 1.

{d1(0),…,d1(5)}	{d1(6),…,d1(9)}	{d1(10),..,d1(15)}	{d1(16),…,d1(24)}	{d1(25),…,d1(35)}	{d1(36),…,d1(44)}	{d1(45),…,d1(49)}	{d1(50),…,d1(57)}	{d1(58),…,d1(74)}	{d1(75),…,d1(94)}	{d1(95),…,d1(108)}	{d1(109),…,d1(127)}
{d2(0),…,d2(8)}	{d2(9),…,d2(15)}	{d2(16),…,d2(19)}	{d2(20),…,d2(22)}	{d2(23),…,d2(33)}	{d2(34),…,d2(44)}	{d2(45),d2(46)}	{d2(47),…,d2(52)}	{d2(53),…,d2(58)}	{d2(59),…,d2(63)}		
{d3(0),…,d3(12)}	{d3(13),…,d3(16)}	{d3(17),…,d3(25)}	{d3(26),…,d3(31)}								
{d4(0),…,d4(9)}	{d4(10),…,d4(13)}	{d4(14),d4(15)}									
{d5(0),…,d5(7)}											
{d6(0),..,d6(3)}											
{d7(0),d7(1)}											
{d8(0)}											
{a8(0)}											

Table 1 - Feature Cluster Boundaries

The design of the feature extractor processor for the software simulation was straightforward. The *FeatureExtractor* class accepts a 256-entry array of 4-bit wavelet coefficients as inputs, and generates a 34-entry array of 4-bit wavelet features. The *FeatureExtractor* class consists of a process function that computes the square root of the sum of the squares of each wavelet coefficient within the predefined boundaries of each feature cluster, as described in equation (9).

7.2 Digital Design Implementation

The digital design of the feature extractor processor consists of a feature extractor module, which contains 34 cluster processors. Each cluster processor performs the operations described in equation (9). The feature extractor module accepts wavelet coefficients as inputs, which it allocates to the cluster processors according to the cluster boundaries. The feature extractor processor outputs 34 4-bit wavelet features. Block diagrams of this system are shown in Figures 9 and 10.

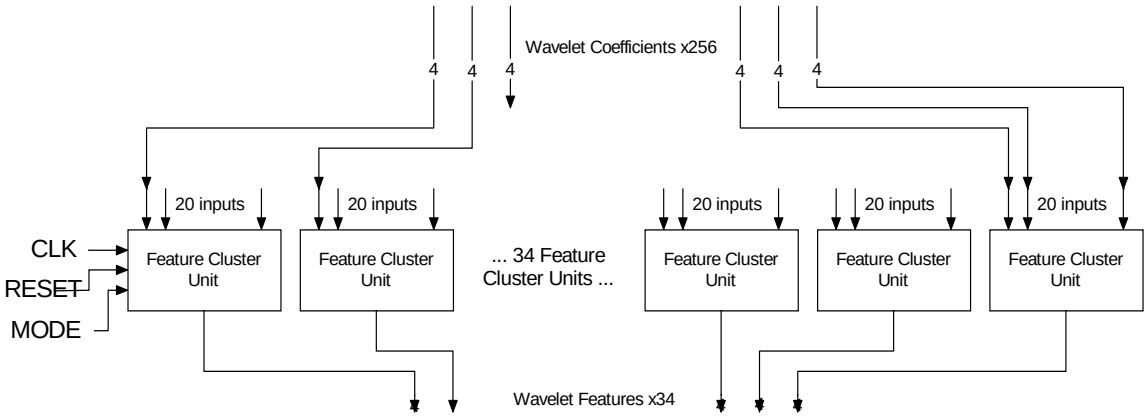

Figure 9 - Feature Extractor Processor

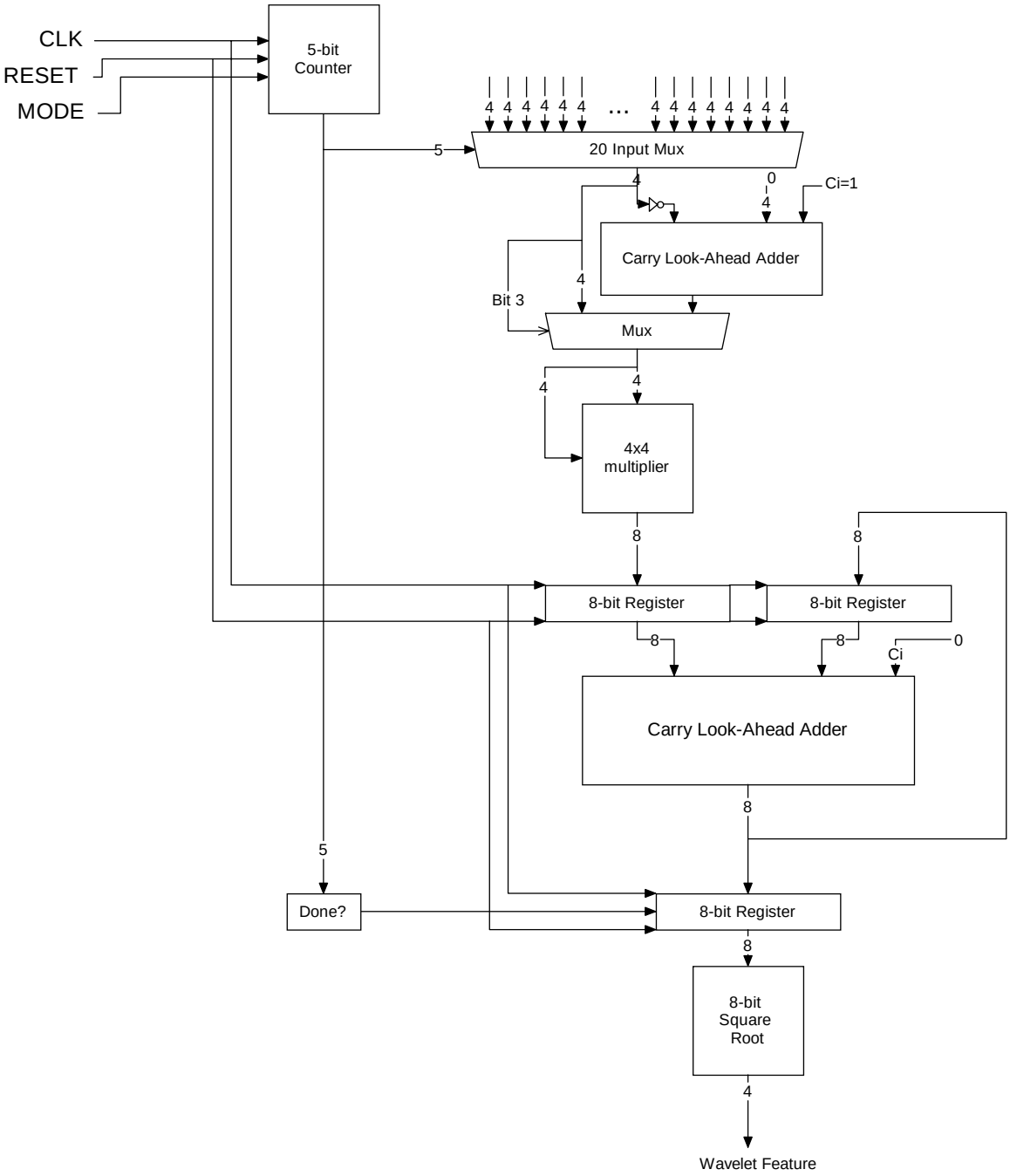

Figure 10 - Feature Cluster Unit

Chapter 8 Design Implementation of a Neural Network

The neural network considered for this thesis was a 2-layer multi-layer perceptron consisting of 34 input neurons, and 2 output neurons. The 34 wavelet-features were fully connected to the 34 input neurons, which were in turn, fully connected to the output neurons. Each output neuron corresponds to each of the two result classes, voice and music.

8.1 Software Simulation

The design of the neural network processor for the software simulation consists of a *NeuralNetwork* class and a *Neuron* class. The *NeuralNetwork* class accepts a 34-entry array of 4-bit wavelet features as inputs, and generates a result classification of "voice", "music", or "other". The *NeuralNetwork* class primarily consists of 34 input *Neuron* objects, 2 output *Neuron* objects, a training function, and a testing function. The *Neuron* class primarily consists of an input array, a weights array, an Activate function, and an output. The *Neuron* Activate function performs the operations in Equations (13) and (14) on the input array to generate the output.

The *NeuralNetwork* training function is called to determine the best values of the weights in each of the neurons for a given training data set. The neural network parameters were determined experimentally beginning with typical values [15]. Given the 8-bit weight limitation imposed on the system, a novel approach was taken in the design of the training algorithm. The training algorithm performs the operations in Equations (15) through (20), by first applying the training set of wavelet features to the neurons, and

calculating the desired response. The training algorithm then updates the weights in each neuron according to the difference between the response from the neurons, and the desired response. If during the training process, the weights of a neuron exceed the bounds of an 8-bit value (-128 to 127) before convergence is reached, all of the weights for that neuron, and for each neuron in its layer are reduced by half. By applying this reduction across the entire layer, the relative strengths of each weight for each neuron is maintained, while allowing for further fine tuning of the weights, resulting in eventual convergence. This reduction by half algorithm generated far better results than when the weights were allowed to grow unbounded and then scaled to 8-bit resolution after convergence. After training, the weights from each neuron can be downloaded to a file.

The *NeuralNetwork* testing function classifies inputs once each neuron's weights have been established by training. The weights can also be uploaded from a file. The testing function is called with a test data set of wavelet features. The test data set is processed by the input neurons, whose results are propagated to the output neurons. The results of the output neurons determine the final classification for the input set. If the first output neuron has the largest output, then the classification is "voice". If the second output neuron has the largest output, then the classification is "music". If both output neurons have an output of zero, then the classification is "other".

8.2 Digital Design Implementation

The digital design of the neural network was inspired by the design in [25]. The weights in the neural network were designed to be uploaded, to avoid the need for training

hardware in the design. The training was instead performed in a software simulation model. To restrict the hardware design to a reasonable size, the neural network data paths were limited to 4-bit wide busses, and the weights were limited to 8-bit values. The neural network processor consists of a main neural network module, which contains 34 input neuron modules, 2 output neuron modules, synchronization hardware to ensure proper data flow through the model, and a result generator module. The result generator observes the outputs of the output neurons, and generates `VALID`, `VOICE`, `MUSIC`, and `OTHER` signals. If the first output neuron's output is greater than or equal to the second's, than the result is "voice". If the second output neuron's output is greater than the first's, the result is "music". If both output neurons' outputs are equal to zero, then the result is "other". The input and output neuron modules were similar in design. The neuron modules contain 8-bit registers to store weights, muxes to select input and weight combinations, and an 8x4 bit multiplier with an accumulator to implement the activation potential calculation. The input and output neurons implement individual activation functions, which are look-up based comparators. A block diagram of this system is shown in Figures 11 and 12. A block diagram of the 8x4-bit multiplier is shown in Figure 13.

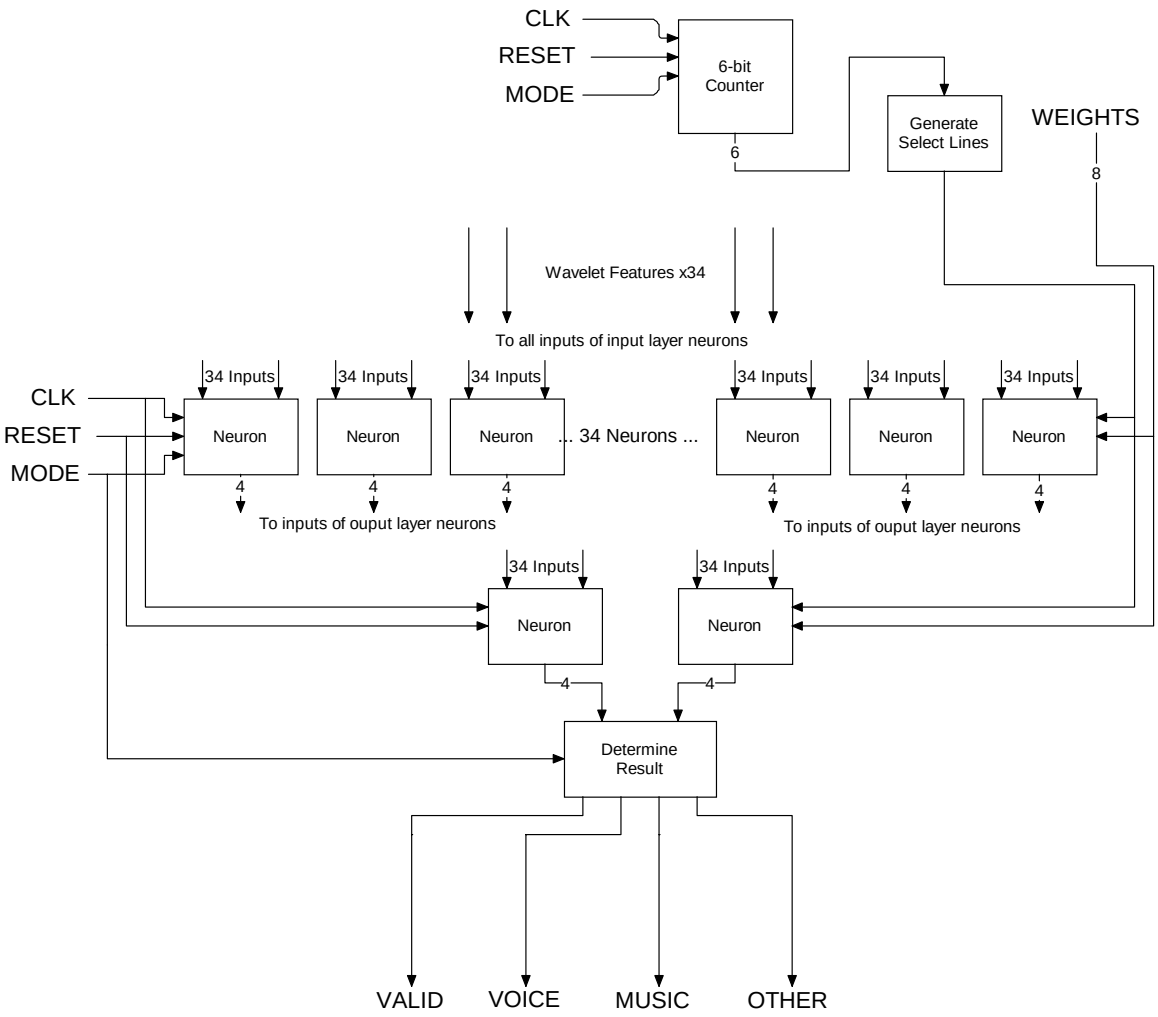

Figure 11 - Neural Network Processor

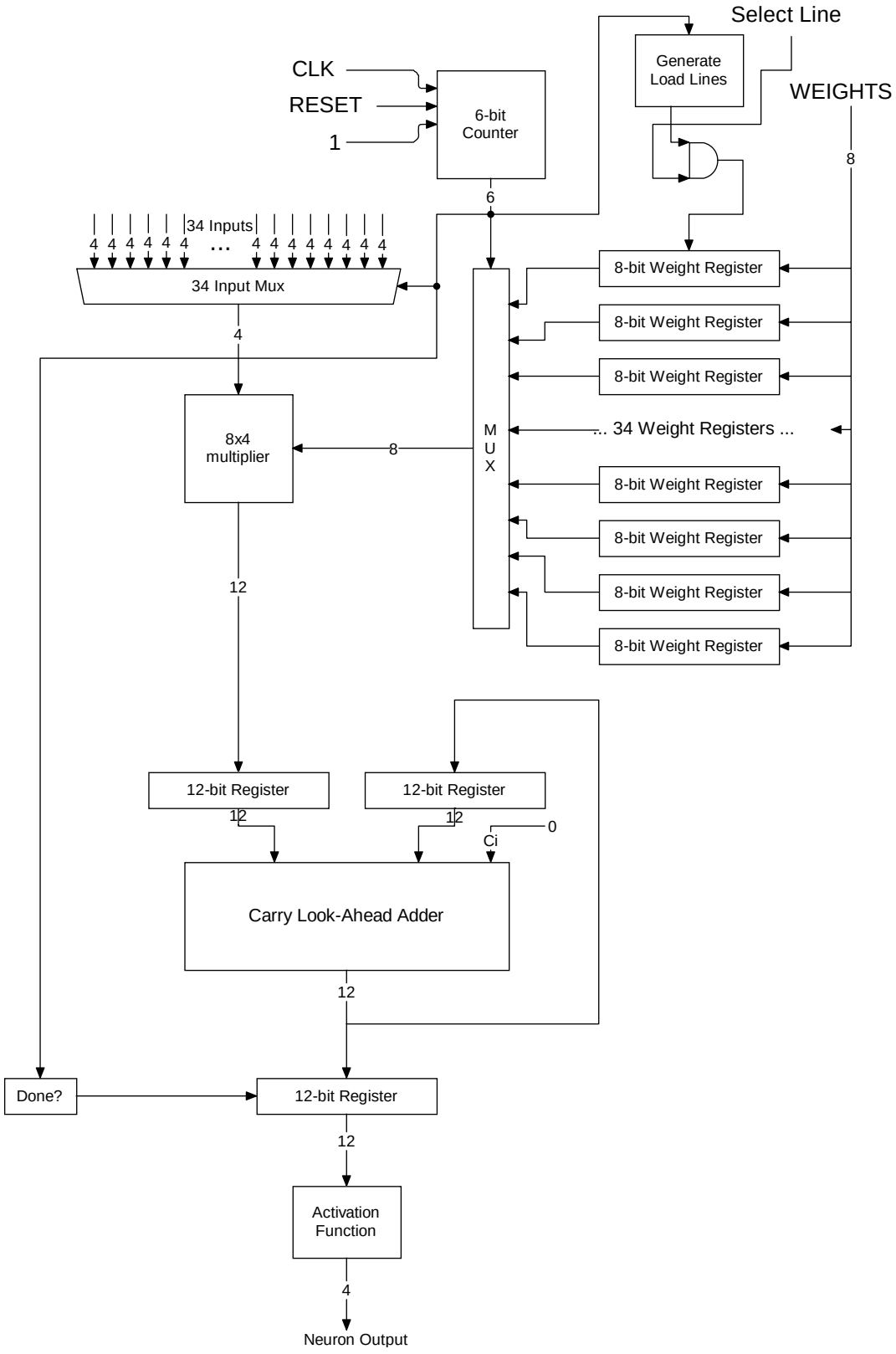

Figure 12 - Neuron

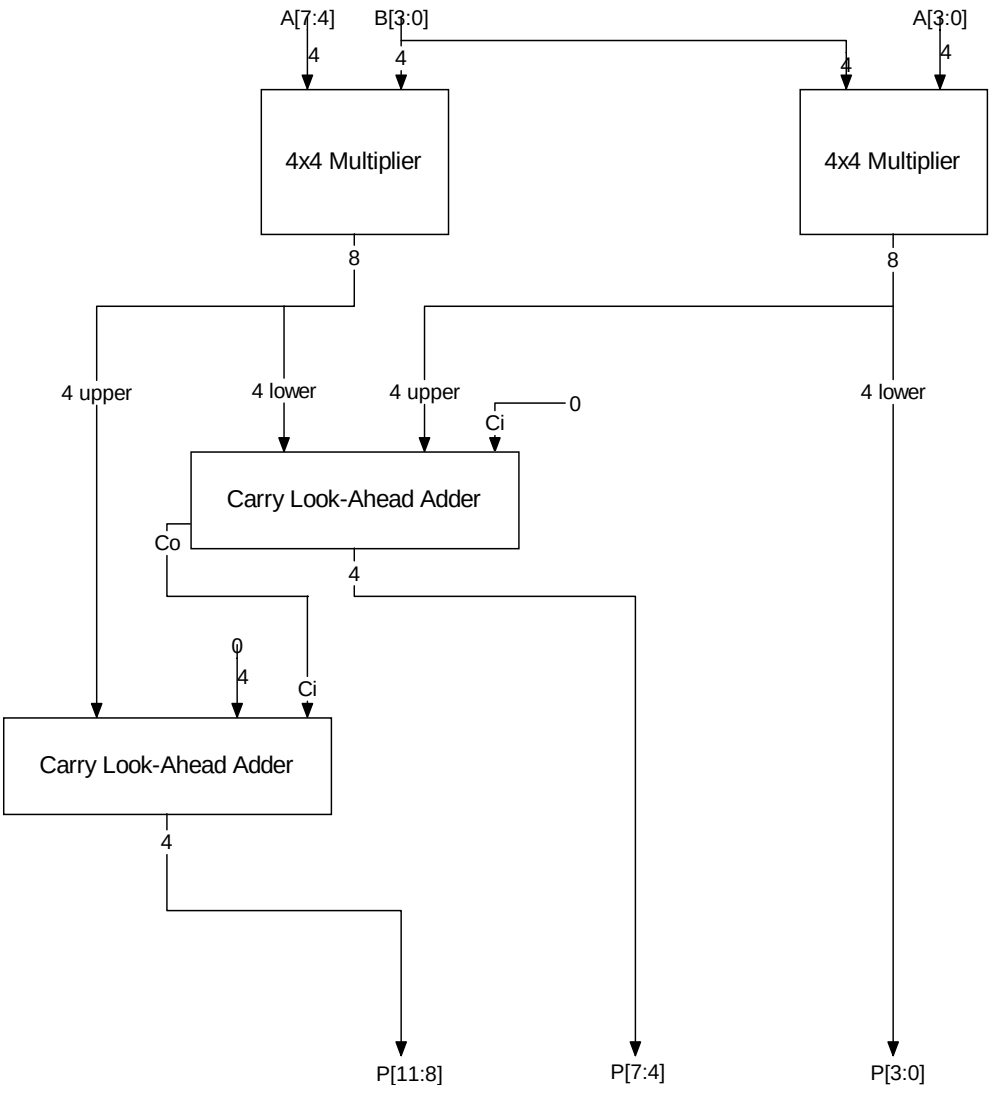

Figure 13 - 8x4-bit Multiplier

Chapter 9 Design Implementation of a Wavelet Neural Network

Wavelet Neural Networks are essentially neural structures with wavelet filters applied to extract features of the input set in the time-frequency domain. Since wavelet filters and neural networks can each take many forms, the combinations and configurations of wavelet neural networks are practically unlimited. The domain of the problem of consideration is the classification of digital audio into voice and music classes. Therefore, a discreet Haar transform with a feature extraction mechanism was combined with a multi-layer perceptron neural network to form a wavelet neural network.

The wavelet neural network considered for this thesis was a discreet Haar wavelet processor with a feature extractor processor combined with a 2-layer multi-layer perceptron consisting of 34 input neurons, and 2 output neurons. The 256 samples of 16-bit digital audio were first applied to the wavelet processor and thereby converted to 256 4-bit wavelet coefficients. The wavelet coefficients were then applied to the feature extractor processor, resulting in 34 wavelet-features. These wavelet-features were fully connected to the 34 input neurons, which were in turn, fully connected to the output neurons. Each output neuron corresponds to each of the two result classes, voice and music.

9.1 Software Simulation

The design of the wavelet neural network for the software simulation was straightforward. The software consisted of a *Wavelet* object, a *FeatureExtractor* object, a *NeuralNetwork* object, and a *TestVector* object. First, the program read the weights file, and uploaded the stored weight to the *NeuralNetwork* object. The program then read the digital audio input file in 256 sample increments. The 256 samples were applied as inputs to the *Wavelet* object, processed, and the results collected in a 256-entry wavelet coefficient array. The 256 wavelet coefficients were applied as inputs to the *FeatureExtractor* object, processed, and the results collected in a 34-entry wavelet feature array. A *TestVector* object was then created from the 34 wavelet features. The *TestVector* object was applied as an input to the *NeuralNetwork* object, processed, and the result was collected to an output file.

9.2 Digital Design Implementation

The digital design of the wavelet neural network includes the wavelet processor, feature extractor processor, neural network processor, and synchronization hardware to ensure proper data flow between the functional blocks. From the top-level hierarchy, as shown in Figure 14, the data inputs to the system are as follows: 16-bit digital audio, which is loaded serially in 16-bit blocks on the rising edge of the clock during classification mode, and 8-bit weights, which are loaded serially in 8-bit blocks on the rising edge of the clock during upload mode. The control signal inputs to the system are as follows: clock signal, `CLK`, which clocks the state machines and registers in the system, an active low reset signal, `RESET`, used to reset all logic, and a mode signal, `MODE`, used to control the

systems to upload weights when `MODE` = ‘0’, or classify audio inputs when `MODE` = ‘1’. The output signals from the system are as follows: `VALID`, used to indicate when the other output signals are valid, and not in transition, `VOICE`, used to indicate the current audio sample has been classified as a voice sample, `MUSIC`, used to indicate the current audio sample has been classified as a music sample, and `OTHER`, used to indicate the current audio sample was not able to be classified as a music or a voice sample.

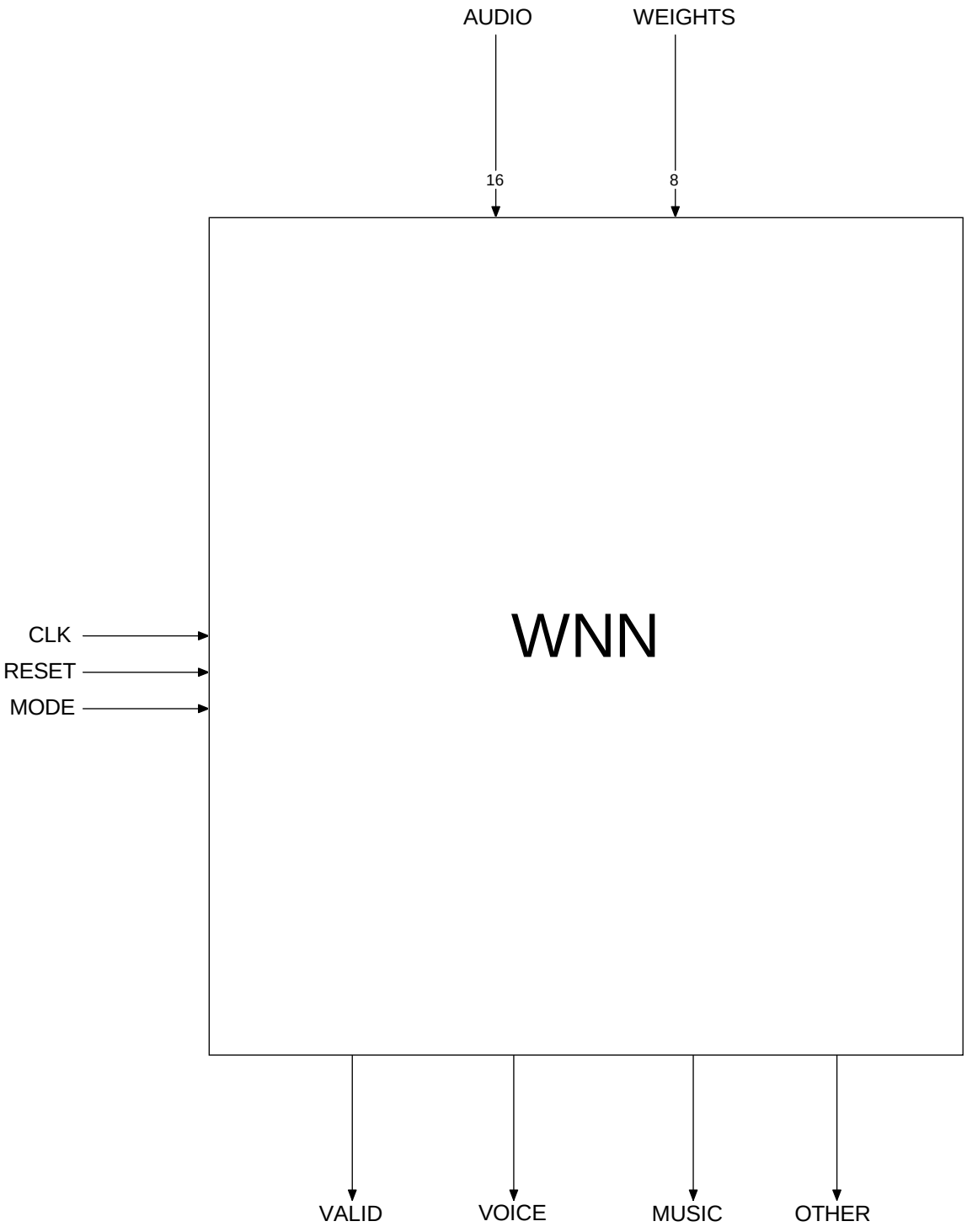

Figure 14 - Wavelet Neural Network Top Level

Within the top-level hierarchy are the processor function blocks and a state machine to control the loading of the weights and audio samples and the propagation of data between blocks, as show in Figure 15. When in upload weights mode, the 8-bit weights are loaded into a register on every clock, and distributed to each weight register of each neuron. This takes 1224 clock cycles to complete (34 weight registers for each of the 34 input neurons, and 34 weight registers for each of the 2 output neurons = 34x34+34x2 = 1224). When in classification mode, the 16-bit audio samples are loaded into a register on every clock, and are input to the wavelet processor. The wavelet processor takes 263 clock cycles to complete the processing of 256 audio samples. This is due to the overhead of synchronizing the high- and low-pass filters with the intermediate capture registers. The wavelet processor is capable of processing audio samples continuously, however, because of save registers used to hold the previous 256 wavelet coefficients. After the wavelet coefficients have been produced by the wavelet processor, they are then input to the feature extractor processor. The feature extractor processor takes 21 clock cycles to complete processing of the wavelet coefficients, because there are a maximum of 20 wavelet coefficients processed by each cluster, plus 1 output register. After the wavelet features have been produced by the feature extractor processor, they are then input to the neural network processor. The neural network processor takes 72 clock cycles to complete, because there are 34 input-weight multiplications iteratively summed, plus 2 output registers for each layer of the neural network (34+2+34+2=72). The wavelet processor continuously runs, once in classification mode, such that after an additional 159 clock cycles after the completion of the neural network processor, the wavelet processor will have already produced another set of 256 wavelet coefficients

ready for feature extraction. The feature extractor begins processing the wavelet coefficients after this time, and the process repeats.

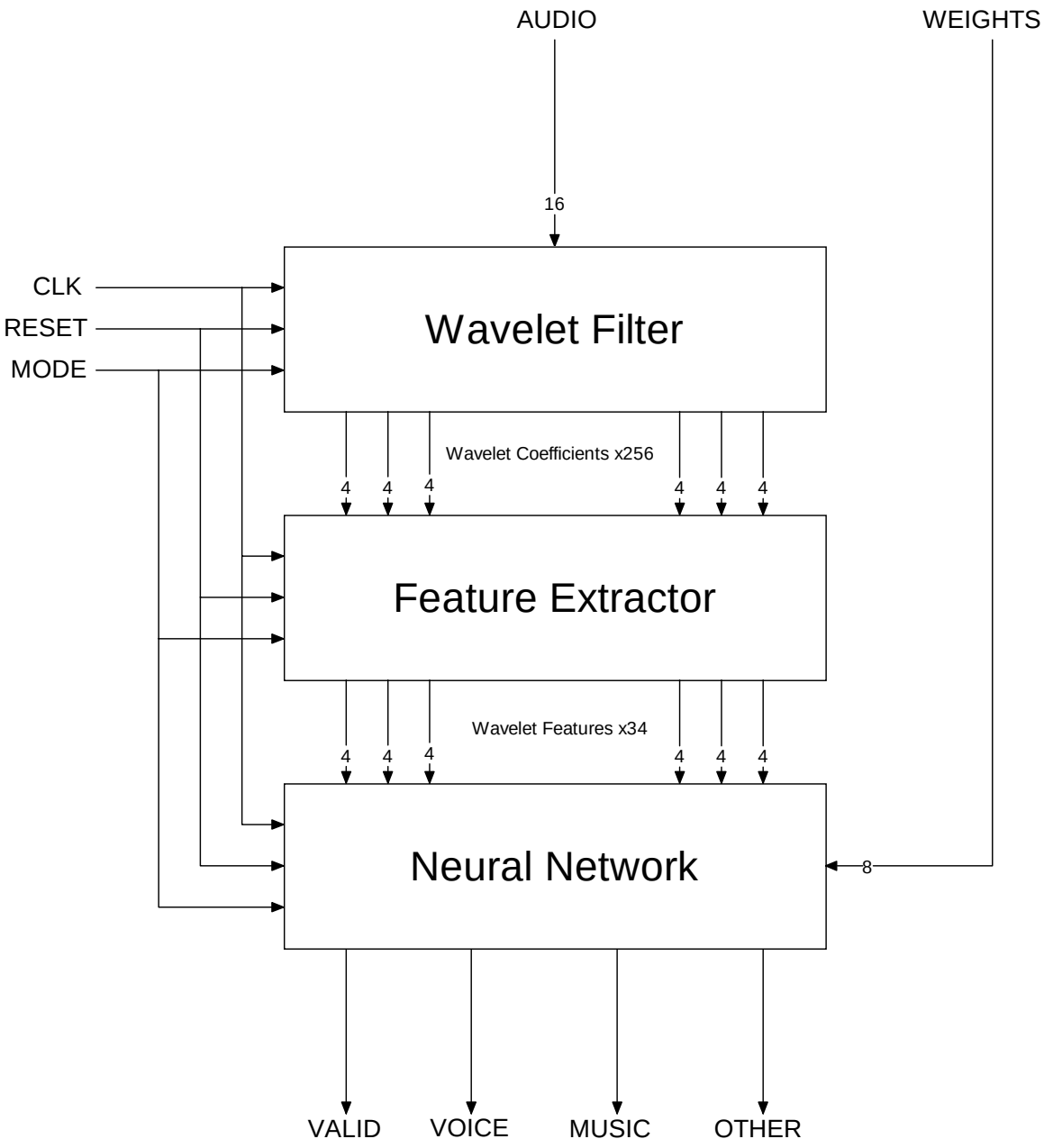

Figure 15 - Wavelet Neural Network Components

Chapter 10 Simulation Results Comparison between Ideal Software and Digital Design Implementation

The wavelet neural network was originally constructed as a software model in order to experiment with network parameters, to determine neural network weight values, to determine ideal results, and to provide a reference to verify correct hardware operation. Using 8-bit weights and 4-bit data paths in the digital design significantly reduced the accuracy of the classifier from the ideal. An 8-bit weights and 8-bit data path model was created in software to demonstrate the capabilities of a larger system. The results of tests performed on the ideal software model and on the hardware models are shown in Table 2. The digital design of the wavelet neural network was written in VHDL and synthesized with Synplicity Synplify, using Actel ProASICPlus APA600 synthesized library cells with a target clock frequency of 11.025 KHz, to match the sampling rate of the digital audio. The results of the synthesis indicated that the design could operate at 15.6 MHz, and required 96,265 logic cells.

	Voice	Music	Other
Ideal Model			
Voice Training Set	98.76%	0.00%	1.24%
Voice Test Set	91.93%	2.88%	5.19%
Music Training Set	0.39%	98.26%	1.35%
Music Test Set	2.49%	90.23%	7.28%
8-bit weights 8-bit data			
Voice Training Set	88.59%	7.79%	3.61%
Voice Test Set	86.74%	10.95%	2.31%
Music Training Set	13.44%	80.85%	5.71%
Music Test Set	10.15%	86.40%	3.45%
8-bit weights 4-bit data			
Voice Training Set	80.23%	17.68%	2.09%
Voice Test Set	70.61%	28.82%	0.58%
Music Training Set	20.79%	77.47%	1.74%
Music Test Set	22.61%	76.44%	0.96%

Table 2 - Classification Results of WNN Configurations

Part III Conclusion

Chapter 11 Conclusion

The trained digital design wavelet neural network was effective in correctly classifying the test data sets. The novel design of the wavelet transform processor produced an efficient hardware design that was also a high performance pipeline. The novel neural network training algorithm was effective in determining weight values that produced excellent classification results. The design of the hardware modules was straightforward to model in VHDL, and the synthesis was simplistic due to the low clock operating speed. The ideal model of the wavelet neural network demonstrates what can be achieved with much larger hardware sizes.

The resulting constrained wavelet neural network processor was capable of correctly classifying test data sets with greater than 70% accuracy. Additional modeling showed that with a reasonable increase in hardware size, greater than 86% accuracy is attainable. The hardware implementation of the wavelet neural network offers advantages over a purely software implementation. The hardware implementation requires a clock rate of only 11.025 kHz to classify audio in real time, whereas an equivalently performing software implementation would require a very fast processor. The hardware implementation also lends well to an embedded design, with one possible application of scanning for talk or music stations in a personal or automobile radio.

This work was published in the Proceedings of the 14th Annual IEEE International ASIC/SOC Conference, 2001 [26].

Chapter 12 Future Work

The most obvious area for future work in this thesis is to implement the digital design in an FPGA with supporting circuitry on a PCB. This would involve laying out the synthesized design, generating a program file, and programming the FPGA. A circuit board containing the FPGA, clocking and reset circuitry, and power, ground, and signal connectors would also have to be constructed. This work was regarded to be beyond the scope of the current thesis, but would be an excellent demonstration of the technology.

Additional avenues of future work include enhancements to the current design. Hardware and logic supporting the back-propagation error-correction learning algorithm could be added to the design, to allow for training the system in a stand-alone environment, which would therefore not require pre-calculating the weights and uploading them to the system. There is room for experimentation in the configuration of the neural network, and the choice of the wavelet filter and feature extraction method. One possible area of experimentation would be to feedback the result of the last classification to the neural network. Since in an audio stream, it is likely that the current sample is of the same classification as the last sample, sporadic errors could possibly be avoided. The system could also be made to be more robust by training the neural network with many more types of music and voice types, than those used in this thesis.

Future areas of possible research include speech recognition, speaker recognition, and content based music genre classification.

Bibliography

[1] Andersen, L.N.; Au, W.; Larsen, J.; Hansen, L.K., Discrimination of cylinders with different wall thicknesses using neural networks and simulated dolphin sonar signals. *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, vol., no.pp.477-486, Aug 1999.

[2] Stefan Pittner and Sagar V. Kamarthi. Feature Extraction From Wavelet Coefficients for Pattern Recognition Tasks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 1, pp. 83-88, Jan. 1993.

[3] Graps, A.L.; An Introduction to Wavelets, *IEEE Computational Sciences and Engineering*, Volume 2, Number 2, Summer 1995, pp 50-61.

[4] W. Sweldens and R. Piessens, Wavelet sampling techniques, *1993 Proceedings of the Statistical Computing Section*. American Statistical Association, 1993, pp. 20-29.

[5] Schleicher, C.; An Introduction to Wavelets for Economists, *Bank of Canada, Working Paper*, No. 2002-3.

[6] Aamir, K.M.; Maud, M.A.; Loan, A., On Cooley-Tukey FFT method for zero padded signals. *Emerging Technologies, 2005. Proceedings of the IEEE Symposium on*, vol., no.pp. 41- 45, 17-18 Sept. 2005.

[7] Kin-Pong Chan and Wai-Chee Fu. Efficient Time Series Matching by Wavelets. In *Proceedings of the 15th International Conference on Data Engineering*, 1998.

[8] Roberto Kawakami, Harrop Galvão, and Takashi Yoneyama. Neural Classifier Employing Biased Wavelets. In *5th Brazilian Symposium on Neural Networks*, 1998.

[9] S.R. Subramanya and Abdou Youssef. Wavelet-based Indexing of Audio Data in Audio/Multimedia Databases. In *Proceedings of the 1998 International Workshop on Multimedia Database Management Systems*, 1998

[10] J. Fridman and E. Manolakos. Distributed memory and control VLSI architectures for the 1-D discrete wavelet transform. *IEEE VLSI Signal Processing*, 1994.

[11] Pati, Y.C.; Krishnaprasad, P.S. Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations. In *IEEE Transactions on Neural Networks*, 1993.

[12] Z.R. Struzik, A. Siebes, Wavelet Transform in Similarity Paradigm II, *CWI Report*, INS-R9815, CWI, Amsterdam, 1998.

[13] Z. Struzik and A. Sibes. Measuring time series similarity through large singular features revealed with wavelet transformation. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications*, pg. 162-166, 1999.

[14] S.G. Mallat, S. Zhong, Complete Signal Representation with Multiscale Edges, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, pp. 710-732, 1992.

[15] Haykin, S. Neural Networks: A Comprehensive Foundation. Macmillan Publishing Company, pp. 10-156, 1994.

[16] Antonio Pedro Timoszczuk and Euvaldo F. Cabral Jr. RBF Neural Networks and MTI for Text Independent Speaker Identification. In *5th Brazilian Symposium on Neural Networks*, 1998.

[17] Adami, A.G.; Lazzarotto, G.B.; Foppa, E.F.; Couto Barone, D.A. A Comparison between Features for a Residential Security Prototype Based on Speaker Identification with a Model of Artificial Neural Network. In *Proceedings of the Third International Conference on Computational Intelligence and Multimedia Applications*, 1999.

[18] Maier, K.D.; Beckstein, C.; Blickhan, R.; Erhard, W.; Fey, D. A Multi-Layer-Perceptron Neural Network Hardware Based on 3D Massively Parallel Optoelectronic Circuits. In *Proceedings of the 6th International Conference on Parallel Interconnects*, 1999.

[19] Rying, E. A., Bilbro, G. L., Lu, J. C. Focused local learning with wavelet neural networks. *IEEE transactions on neural networks*, 13(2), pg. 304-319, 2002

[20] Ho D.W.C, Zhang P-A, and Xu J.: Fuzzy Wavelet Networks for Function Learning, *IEEE Transactions on Fuzzy Systems*, Vol. 9, No.1, pp. 200-211, 2001

[21] B.-L. Zhang, R. Coggins, M.A. Jabri, D. Dersch, and B. Flower. Multiresolution Forecasting for Futuretrading Using Wavelet Decompositions. *IEEE Transactions on Neural Networks*, Vol. 12, pp. 766-775, 2001.

[22] Zhang, Q. Using wavelet network in nonparametric estimation. *Technical Report 833, IRISA*, 1994.

[23] A. B. Geva. ScaleNet -- Multiscale Neural-Network Architecture for Time Series Prediction. *IEEE Transactions on Neural Networks*, Vol. 9, No. 5, pp. 1471-1482, 1998.

[24] Wang, X., Yu, G., and Lee, J. Wavelet Neural Network for Machining Performance Assessment and Its Implications to Machinery Prognostics, *Proceedings of the 5th International Conference on Managing Innovations in Manufacturing*, 2002.

[25] Hoda S. Abdel-Aty-Zohdy and Mahmoud Al-Nsour. Digital Neural Processing Unit for Electronic Nose. In *Proceedings of the Ninth Great Lakes Symposium on VLSI*, 1998.

[26] Hughes, J.; Gaborski, R.; Hsu, K.; Titus, A., An auditory classifier employing a wavelet neural network implemented in a digital design. *ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International*, vol., no.pp.8-12, 2001.